A Broadcasting Protocol for Video-on-Demand Using Optional Partial Preloading

Jehan-François Pâris[1]

Department of Computer Science, University of Houston, Houston, TX 77204-3010
paris@cs.uh.edu

Abstract. Broadcasting protocols for video-on-demand normally require customers to wait for a few minutes before starting to watch the video of their choice. The easiest way to avoid this delay is to preload in each customer set-top-box the first few minutes of each video. We present here a broadcasting protocol making this requirement optional: customers are offered the option of preloading the first few minutes of one or more videos in order to eliminate the small delay they would otherwise experience. We also show that this feature can be implemented at a very reasonable cost.

Keywords: video-on-demand, broadcasting protocol, pagoda broadcasting, zero-delay broadcasting.

1 Introduction

Broadcasting protocols are expected to play an important role in the commercial success of video-on-demand services because they offer the most efficient way of distributing very popular videos to large metropolitan audiences. Unlike other video distribution protocols, broadcasting protocols distribute video contents according to a fixed schedule guaranteeing that all customers will receive these contents on time. As a result, the video server workload is not affected by the number of customers using the service.

All recent broadcasting protocols for video-on-demand derive in some fashion from Viswanathan and Imielinski's *pyramid broadcasting protocol* [10]. Like it, they partition each video into segments that are simultaneously broadcast on different channels. They also require customers to be connected to the service through a "smart" set-top box (STB) capable of (a) receiving data at rates exceeding the video consumption rate and (b) storing locally the video data that arrive out of sequence. Finally, they assume that customers will watch videos in sequential fashion without any fast forwards. This setup allows broadcasting protocols to transmit the various segments of each video using less and less bandwidth as the video progresses. Each video segment will typically require less bandwidth than its predecessor with the initial segment thus requiring the most bandwidth.

[1] Supported in part by the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390.

One major limitation of this approach is that broadcasting protocols cannot provide true instant access to videos without dedicating an inordinate amount of bandwidth to the first few segments of the video. *Partial preloading* [4] solves this problem by preloading in customer STBs the first few minutes of all the videos being broadcast. It provides zero-delay access to these videos while significantly reducing the server aggregate bandwidth.

We propose to extend this approach to cases where we cannot expect all customer STBs to always have received and stored the first few minutes of all videos being offered. *Optional partial preloading* (OPP) assumes instead that there will be two kinds of customers for each video, namely, thOse who have the first few minutes of the video preloaded in their STBs and those who will need to receive the whole video. The protocol will provide instant access to the video to all the customers who have its first few minutes preloaded in their STBs while requiring other customers to wait for at most a few minutes.

As we will see, this flexibility comes with a cost. First, there will be no bandwidth savings, as we now have to broadcast the first few minutes of each video for the sake of the customers not having these minutes preloaded in their STB. Second, the higher quality of service provided to customers who have the first few minutes of the video preloaded in their STBs will require more frequent transmissions of some video segments and will result in an increase of the waiting time for the other customers. We found that the best way to control this increase is to combine optional partial preloading with a fixed-delay broadcasting protocol such as the fixed-delay pagoda broadcasting protocol (FDPB) [5]. For instance, an FDPB protocol allocating five broadcasting channels to each video can provide zero-delay access to all customers who preloaded the first two minutes and half of the video while requiring other customers to wait 97 seconds.

The remainder of this paper is organized as follows. Section 2 reviews relevant previous work on broadcasting protocols. Section 3 discusses optional partial preloading and introduces our FDPB protocol with optional partial preloading. Section 4 evaluates its performance and Section 5 has our conclusions.

2 Previous Work

Given the large number of video broadcasting protocols that have been proposed since Viswanathan and Imielinski's *pyramid broadcasting* protocol, we will only mention here those protocols that are directly relevant to our work. The reader interested in a more comprehensive review of broadcasting protocols for video-on-demand may want to consult reference [1].

The simplest broadcasting protocol is Juhn and Tseng's *fast broadcasting* (FB) protocol [2]. The FB protocol allocates to each video k data channels whose band-widths are all equal to the video consumption rate b. It then partitions each video into 2^{k-1} segments, S_1 to $S_{2^{k-1}}$, of equal duration d. As Figure 1 indicates, the first channel continuously rebroadcasts segment S_1, the second channel transmits segments S_2 and S_3, and the third channel transmits segments S_4 to S_7. More generally, channel j with $1 \leq j \leq k$ transmits segments $S_{2^{j-1}}$ to S_{2^j-1}.

First Channel	S_1	S_1	S_1	S_1
Second Channel	S_2	S_3	S_2	S_3
Third Channel	S_4	S_5	S_6	S_7

Figure 1. The first three channels for the FB protocol.

First Channel	S_1	S_1	S_1	S_1	S_1	S_1
Second Channel	S_2	S_4	S_2	S_5	S_2	S_4
Third Channel	S_3	S_6	S_8	S_3	S_7	S_9

Figure 2. A PB protocol with three channels

When customers want to watch a video, they wait until the beginning of the next transmission of segment S_1. They then start watching that segment while their STB starts downloading data from all other channels. Hence the maximum customer waiting time is equal to the duration of a segment. Define a *slot* as a time interval equal to the duration of a segment. To prove the correctness of the FB protocol, we need only to observe that each segment i with $1 \le i \le 2^k - 1$ is rebroadcast at least once every i slot. Then any client STB starting to receive data from all broadcasting channels will always receive all segments on time.

The *pagoda broadcasting* (PB) [3] protocol improves upon the FB protocol by using a more complex segment-to-channel mapping. As seen on Figure 2, the PB protocol can pack nine segments into three channels while the FB protocol can only pack seven of them. Hence the segment size will be equal to one ninth of the duration of the video and no customer would ever have to wait more than 14 minutes for a two-hour video.

Neither the FB protocol nor the PB protocol require customer STBs to wait for any minimum amount of time. As a result, there is no point in requiring customer STBs to start downloading data while customers are still waiting for the beginning of the video. The newer *fixed-delay pagoda broadcasting* (FDPB) protocol [5] requires all users to wait for a fixed delay w before watching the video they have selected. This waiting time is normally a multiple m of the segment duration d. As a result, the FDPB protocol can partition each video into much smaller segments than either FB or PB with the same number of channels. Since these smaller segments can be packed much more effectively into the k channels assigned to the video, the FDPB protocol achieves smaller customer waiting times than FB and PB protocols with the same number of channels.

Figure 3 summarizes the segment-to-channel mappings of a FDPB protocol requiring customers to wait for exactly 9 times the duration of a segment. Since customers have to wait for 9 times that duration, the first segment of the video will need to be broadcast at least once every 9 slots. Hence the protocol will use time division multiplexing to partition the first channel into $\sqrt{9}$ subchannels with each subchannel containing one third of the slots of the channel. The first subchannel will continuously broadcast segments S_1 to S_3 ensuring that these segments are repeated exactly once every 9 slots.

Channel	Subchannel	First Segment	Last Segment
C_1	1	S_1	S_3
	2	S_4	S_7
	3	S_8	S_{12}
C_2	All 5 subchannels	S_{13}	S_{42}
C_3	All 7 subchannels	S_{43}	S_{116}
C_4	All 11 subchannels	S_{117}	S_{308}
C_5	All 17 subchannels	S_{293}	S_{814}

Figure 3. Segment to channel mappings of a FDPB protocol with $m = 9$.

Observe that the next segment to be broadcast, segment S_4 needs to be broadcast once every 12 slots. Hence the second subchannel will transmit segments S_4 to S_7 ensuring that these segments are repeated exactly once every 12 slots. In the same way, the third subchannel will broadcast segments S_8 to S_{12} ensuring that these segments are repeated exactly once every 15 slots.

The process will be repeated for each of the following channels partitioning each channel into a number of subchannels equal to the square root of the minimum periodicity of the lowest numbered segment to be broadcast by the channel. Hence channel C_2 will be partitioned into 5 subchannels because segment S_{13} needs to be repeated every 21 slots and $\sqrt{21} \approx 5$. As a result, the protocol will map segments S_{13} to S_{42} into the 5 subchannels of the second channel. Repeating the same process on channels C_3 to C_5, the protocol will be able to map 814 segments into five channels and achieve a deterministic waiting time of 9/814 of the duration of the video, that is, 80 seconds for a two-hour video.

3 Optional Partial Preloading

Recall that all recent broadcasting protocols require customers to be linked to the service through a STB capable of storing locally the video data that arrive out of sequence. In the current state of storage technology, this implies having a disk drive in each STB, a device already present in the so-called digital VCR's offered by TiVo [8], Replay [6] and Ultimate TV [9]. Today, the disk drives used in the cheapest computers have capacities of at least 10 gigabytes, giving them the possibility of storing at least three hours and half of video in MPEG-2 format.

Partial preloading [4] uses some of this disk capacity to preload, say, the first 6 minutes of the top 10 videos or the first 3 minutes of the top 20 videos. The technique offers two major advantages. First it will provide instant access to these videos. Second, it will reduce the bandwidth required to broadcast them as the first minutes of each video could be broadcast much less frequently.

Channel	Subchannel	First Segment	Last Segment
C_1	1	S_{10}	S_{12}
	2	S_{12}	S_{16}
	3	S_{17}	S_{21}
C_2	All 5 subchannels	S_{22}	S_{51}
C_3	All 7 subchannels	S_{52}	S_{125}
C_4	All 11 subchannels	S_{126}	S_{317}
C_5	Not used		

Figure 4. Segment to channel mappings of a FDPB protocol preloading the first 9 segments of a video.

Consider for instance the FDPB protocol discussed in the previous section and assume that we preload in the customer STB the first 9 segments of each video. The first segment of the video that will need to be broadcast will be segment S_{10} and this segment will need to be broadcast at least once every 9 slots. As shown on Figure 4, the protocol will partition the first channel into $\sqrt{9}$ subchannels with the first subchannel broadcasting segments S_{10} to S_{12}, the second subchannel transmitting segments S_{13} to S_{16} and the third subchannel broadcasting segments S_{17} to S_{21}. As one can see, this mapping is almost identical to the mapping displayed on Figure 3, the sole difference being that the first segment to be broadcast is now segment S_{10} instead of segment S_1.

Allocating four channels to the video would allow us to partition the video into 311 segments and offer zero-delay access to that video while requiring all customers to preload 9/317 of it in their STB, that is, a little bit less than two minutes and a half of video data for a two-hour video. We could thus provide a better quality of service while using one less channel. A much more aggressive partial preloading policy could only assign three channels to the video and require customers to preload 9/125 of each video in their STB, that is, less than nine minutes of video data for a two-hour video.

The sole problem with partial preloading is that customers who do not have the first few minutes of a video preloaded in their STB will not be able to watch that video. Consider, for instance, the case of customers having just installed their “smart” set-top box and wanting to watch a video. They would have to wait for up to one hour before being able to watch the video of their choice. Similar delays would be experienced by anyone not willing to have his or her STB permanently turned on.

We propose here a more flexible solution: customers who accept to preload in their STB the first few minutes of selected videos will get true instant access to them while other customers will be able to access the video after waiting for a few minutes. This approach has two great advantages. First, it motivates customers to accept partial preloading. Second, it takes care of the customers who have not preloaded these first few minutes.

First Channel	S_1	S_1	S_1	S_1	S_1	S_1
Second Channel	S_2	S_2	S_2	S_2	S_2	S_2
Third Channel	S_3	S_5	S_3	S_6	S_3	S_5
Fourth Channel	S_4	S_7	S_9	S_3	S_8	S_{10}

Figure 5. A PB protocol with four channels offering instant access to the customers having chosen to preload the first segment of the video in their STB.

Number of Channels	Number of Segments	
	Original PB Protocol	PB protocol with OPP
1	1	1
2	3	2
3	9	4
4	19	10
5	49	20
6	99	50
7	249	100

Figure 6. Comparing the number of segments broadcast by the original PB protocol and the PB protocol with optional partial preloading (OPP) over the same number of channels.

The sole problem with making partial preloading *optional* is its cost. Consider the case of a Pagoda Broadcasting protocol where customers can chose to preload the first segment of the video. To ensure on time delivery of all video data, the second segment must be broadcast in such a way its contents can be received while the customer is watching the preloaded part of the video. Hence segment S_2 must be repeated once every slot. By induction on i, segment S_i with $i \geq 2$ will have to be repeated at least once every $i - 1$ slots.

Figure 5 summarizes the resulting segment-to-channel mapping for the first four channels. Comparing with the mappings of the original PB protocol in Figure 2, we see that our new PB protocol would require 4 channels instead of 3 to broadcast 10 segments instead of 9. More generally, the protocol would always require one additional channel to broadcast one additional segment. As seen on Figure 6, we would need one extra channel to ensure that customers who do not preload wait the same amount of time as before.

A better solution is to use a modified FDPB protocol. Consider for instance the case of a FDPB protocol offering instant access to the customers having preloaded the first 12 segments of the video while requiring other customers to wait for 9 times the duration of a segment. As shown on Figure 7, the segment mappings for the first channel are identical to those of a FDPB protocol without optional partial prefetching: the channel will broadcast the 12 first segments of the video over three subchannels.

Channel	Subchannel	First Segment	Last Segment
C_1	1	S_1	S_3
	2	S_4	S_7
	3	S_8	S_{12}
C_2	1	S_{13}	S_{15}
	2	S_{16}	S_{18}
	3	S_{19}	S_{22}
	4	S_{23}	S_{27}
C_3	All 5 subchannels	S_{28}	S_{64}
C_4	All 8 subchannels	S_{65}	S_{162}
C_5	All 13 subchannels	S_{163}	S_{414}

Figure 7. Segment to channel mappings of a FDPB protocol with $m = 9$ offering instant access to the customers having preloaded the first 12 segments of the video.

The situation is quite different for the other channels. The first segment to be broadcast by the second channel is segment S_{13}. As before, customers who have not preloaded the first 12 segments of the video will start viewing S_{13} after (a) having waited for 9 times the duration of a segment and (b) having watched the first 12 segments of the video. Hence, these segment still need to be repeated at least once every 9 + 12 = 20 slots. This is not true for the customers who had preloaded the first 12 segments of the video: since they have instant access to the video, they need to have segment S_{13} repeated at least once every 12 slots. By induction on i, we find that all segments S_i with $i \geq 12$ will need to be repeated at least once every $i - 1$ slots. We can generalize our observation to the case of an FDPB protocol offering instant access to the customers having preloaded the first n_p segments of the video while requiring other customers to wait for m times the duration of a segment. The protocol will have to repeat each segment S_j

a) at least once every $m + i - 1$ slots if $i \leq n_p$, and
b) at least once every $i - 1$ slots if $i > n_p$.

Figure 7 shows the results of these tighter requirements. Since segment S_{13} has to be repeated at least once every 12 slots, the second channel is now partitioned into 4 subchannels and will only broadcast segments S_{13} to S_{27}, that is, 15 less segments than a FDPB protocol without optional partial preloading. Comparing the mappings displayed in Figure 7 with those displayed in Figure 3, we can see that the same observation applies to all subsequent channels. For instance, a FDPB with optional partial preloading using 5 channels and requiring customers who did not preload the first 12 segments of the video to wait for a time equal to the duration of 9 segments will be able to partition each video into 414 segments, that is, 388 less segments that the original FDPB protocol. Hence, customers who did not preload the first 12/414 of the

Channel	Number of subchannels	First Segment	Last Segment
C_1	10	S_1	S_{156}
C_2	12	S_{157}	S_{400}
C_3	20	S_{401}	S_{1051}
C_4	32	S_{1052}	S_{2787}
C_5	53	S_{2788}	S_{7461}

Figure 8. Segment to channel mappings of a FDPB protocol with m = 100 offering instant access to the customers having preloaded the first 156 segments of the video.

video will have to wait for a time equal to 9/414 of the video duration, that is, about two minutes and half for a two-hour video. Instant access to the video will be provided to the customers who have the 12 first segments of the video preloaded in their STB, that is about three minutes and half of video data for the same two-hour video. In contrast, the original FDPB protocol would have required all customers to wait for 9/802 of the video duration, that is, less than one minute and half for the same two-hour video.

Even better results can be achieved by portioning each video into much smaller segments with resulting increases of m and n_p. Figure 8 shows for instance the segment to slot mappings for a FDPB protocol offering instant access to the customers having preloaded the first 156 segments of the video while requiring other customers to wait for 100 times the duration of a segment. As we can see, a FDPB with optional partial preloading using 5 channels and requiring customers who did not preload the first 156 segments of the video will be able to partition each video into 7461 segments. Hence, customers who did not preload the first 156/7461 of the video will have to wait for a time equal to 100/7461 of the video, that is, 97 seconds for a two-hour video. Instant access to the video will be provided to the customers who have the 156 first segments of the video preloaded in their STB, that is about two minutes and half of video data for the same two-hour video

4 Performance Analysis

In this section, we derive first lower bounds for the bandwidth requirements of fixed-delay broadcasting protocols with and without partial preloading. We present then some data for two FDPB protocols with and without optional partial preloading.

To compute these lower bounds, let us consider a video of duration D being broadcast in such a way that all customers requesting the video wait for w time units before starting the video. Let Δt represent a small time interval starting at a location t within the video. To avoid STB underflow, the contents of this time interval must be broadcast at a minimum bandwidth $b/(t+w)$ where b is the video consumption rate.

Summing over all intervals as Δt approaches 0, we see that the bandwidth required to transmit the video is given by

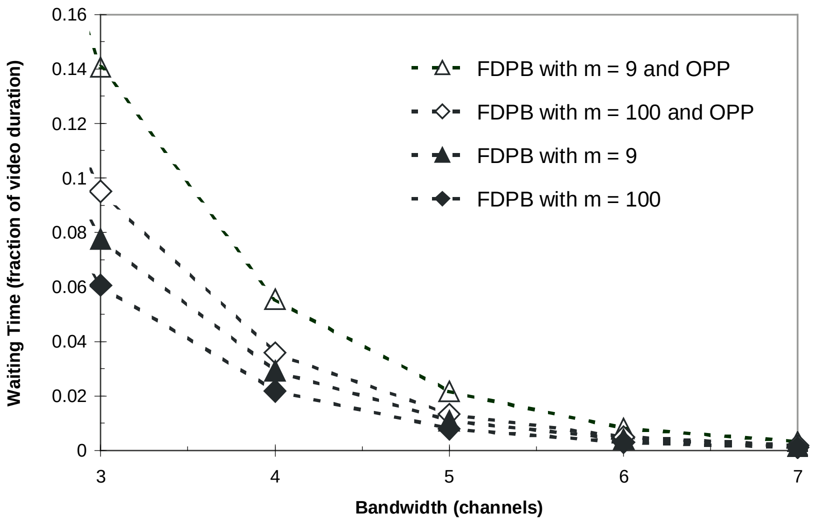

Figure 9. Customer waiting times achieved by FDPB protocols with or without OPP. Note that the customers having preloaded the required number of video segments in their STB have instant access to the video

$$\int_0^D \frac{b}{t+w}dt = b(\ln(D+w) - \ln w) = b\ln\frac{D+w}{w} \tag{1}$$

Assume now that the protocol requires all customers to have the *f* first minutes of the video preloaded in their STB. The minimum bandwidth required to transmit the non-preloaded part of the video would then be given by:

$$\int_f^D \frac{b}{t}dt = b(\ln D - \ln f) = b\ln\frac{D}{f} \tag{2}$$

In most practical cases *w* and f will be much smaller than the video duration *D* and thus $D + w \approx D$. Comparing equations (1) and (2), we can see that the bandwidth required for broadcasting a video of duration *D* with a fixed customer delay *w* is almost equal to the bandwidth required for broadcasting the same video when the customers have the first *w* minutes of that video preloaded in their STBs.

Let us see now what happens when we make the preloading of the *f* first minutes of the video optional and have to broadcast these *f* minutes separately. The minimum bandwidth to distribute these *f* first minutes will be given by:

$$\int_0^f \frac{b}{t+w}dt = b(\ln(w+f) - \ln w) = b\ln\frac{w+f}{w} \tag{3}$$

By adding equations (2) and (3), we obtain the minimum bandwidth required to broadcast the whole video, namely:

$$b\ln\frac{D}{f}b + \ln\frac{w+f}{w} + = b\ln\frac{D(w+f)}{wf} \tag{4}$$

and the extra cost of providing instant video access to the customers who preloaded the f first minutes of the video is then given by:

$$b\ln\frac{D(w+f)}{wf}-b\ln\frac{D+w}{w}=b\ln\frac{D(w+f)}{f(D+w)} \tag{5}$$

Observing that f is a factor of the denominator while w only appears as term, we can see that the best way to reduce the additional cost of optional preloading is to increase the size of the preloaded portion of each video. The main drawback of this approach is the additional storage requirements that a large f implies and the additional demands it makes on the customer STB.

Figure 9 compares the customer waiting times achieved by two FDPB protocols with and without optional partial preloading (OPP). The first FDPB protocol requires customers to wait for 9 times the duration of a video segment ($m = 9$) while the other uses much smaller segments and requires customers to wait for 100 times the duration of a segment ($m = 100$). Bandwidths are expressed in channels. Since all these channels have a bandwidth equal to the video consumption rate b, a bandwidth of seven channels represents seven times the video consumption rate. All waiting times are expressed as fractions of the video duration D. So, a value of 0.05 would correspond to a maximum waiting time of six minutes for a two-hour video.

Several observations can be made from this Figure. First, implementing OPP without increasing the server bandwidth always results in an increase of the waiting time for the customers who have not preloaded the first few minutes of the video. Second, the FDPB protocol using smaller segments and $m = 100$ always provides smaller customer waiting times than the FDPB protocol with $m = 9$. This was expected. The big surprise was that the gap between the waiting times achieved by FDPB protocols with OPP and the waiting times achieved by the original FDPB protocols was strongly affected by m. Adding OPP to an FDPB protocol with $m = 9$ always *doubles* the waiting time for the customer who have not preloaded the video while adding the same option to an FDPB protocol with $m = 100$ only increased the same waiting time by at most 68 percent.

One could thus conclude from this study that using smaller and smaller segments with larger and larger values of m will result in improved performances for the FDPB with and without OPP. There are however two limiting factors to consider. First, increasing the value of m above 100 is not likely to bring much additional benefit as it was found that FDPB with $m = 100$ already performs very close to the theoretical minimum given by equation (1) [5]. Second, partitioning each video into a multitude of very small segments would have a negative effect on the I/O bandwidth of the video server as small reads are inherently less efficient than large reads.

5 Conclusions

Partial preloading offers a very cost effective solution to the problem of offering instant access to popular videos. Unfortunately, it is not very realistic to assume that all customers will always have the first few minutes of every video they want to watch preloaded in their set-top box.

Optional partial preloading (OPP) solves this problem by making preloading optional. It provides customers who have preloaded the first few minutes of a video in their set-top box with instant access to that video while allowing other customers to watch it after a short delay.

We have presented a fixed-delay pagoda broadcasting protocol (FDPB) implementing OPP and shown that implementing OPP without increasing the server bandwidth always resulted in a significant increase of the waiting time for the customers who have not preloaded the first few minutes of the video. We have also found that this increase was significantly affected by the *m* parameter of the FDPB protocol we used: FDPB protocols using large numbers of small segments and large values of *m* performed much better than protocols using large segments and small values of *m*.

Given their deterministic nature and their low server bandwidth requirements, broadcasting protocols like FDPB can be easily implemented on off-the-shelf workstations [7]. Their ultimate success is likely to depend on the customer acceptance of the smart set-top boxes these protocols require.

References

[1] S. W. Carter, D. D. E Long and J.-F. Pâris, Video-on-demand broadcasting protocols, In *Multimedia Communications: Directions and Innovations* (J. D. Gibson, Ed.), Academic Press, San Diego, 2000, pages 179–189.

[2] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44(1):100–105, March 1998.

[3] J.-F. Pâris, S. W. Carter and D. D. E. Long. A hybrid broadcasting protocol for video on demand. *Proc. 1999 SPIE Conference on Multimedia Computing and Networking* (MMCN '99), Jan. 1999, pages 317–326.

[4] J.-F. Pâris, D. D. E. Long and P. E. Mantey. A zero-delay broadcasting protocol for video on demand. *Proc. 1999 ACM Multimedia Conference*, Nov. 1999, pages 189–197.

[5] J.-F. Pâris. A fixed-delay broadcasting protocol for video-on-demand, *Proc. 10th International Conference on Computer Communications and Networks* (ICCCN '01), pages 418–423, Oct. 2001.

[6] ReplayTV. http://www.replay.com/.

[7] K. Thirumalai, J.-F. Pâris and D. D. E. Long. Tabbycat: An inexpensive scalable server for video-on-demand, *submitted for publication*.

[8] TiVo Technologies. http://www.tivo.com/.

[9] UltimateTV. http://www.ultimatetv.com/.

[10] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):197–208, 1996.